Отчёт по лабораторной работе 3

Андрианова Марина Георгиевна

Содержание

[Цель работы 1](#_Toc179633621)

[Задание 1](#_Toc179633622)

[Теоретическое введение 1](#_Toc179633623)

[Используемые функции 2](#_Toc179633624)

[Выполнение лабораторной работы 2](#_Toc179633625)

[Выводы 4](#_Toc179633626)

[Список литературы 4](#_Toc179633627)

Список иллюстраций

[‘Введение функции и запись переменных’ 2](#_Toc179633628)

[‘Создание цикла for с побитовым сложением’ 3](#_Toc179633629)

[‘Функция шифровки и дешифровки’ 3](#_Toc179633630)

[‘Вывод переменных’ 3](#_Toc179633631)

[‘Итоговый код’ 4](#_Toc179633632)

[‘Результат выполнения программы’ 4](#_Toc179633633)

Список таблиц

**Элементы списка иллюстраций не найдены.**

# Цель работы

Изучить шифрование гаммированием и реализовать программу шифрования, используя язык программирования Julia.

# Задание

***Программно реализовать на языке Julia шифрование гаммированием***

# Теоретическое введение

Шифрование гаммированием — это симметричный метод шифрования, при котором на открытый текст накладывается последовательность, сформированная из случайных чисел.

Важным в шифровании гаммированием является то, что сгенерированный случайно ключ должен быть той же длины, что и сообщение, которое необходимо зашифровать.

В данном виде шифрования используется *побитовое сложение*, а данная операция является обратной, поэтому для того, чтобы расшифровать результат нужно просто прогнать его через шифр с тем же самым ключом.

Пример: a = 01000011, b = 01110010 тогда c = a xor b:

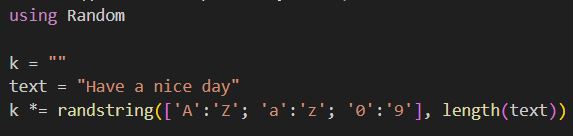
c = 00110001, а чтобы получить обратно a, меняем их местами в в формуле

## Используемые функции

randstring - создает рандомную строку из заданных данных, заданной длины xor - производит побитовое сложение codepoint - возвращает кодовую точку Unicode соответствующую символу

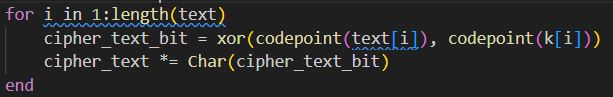
# Выполнение лабораторной работы

Создаю переменные *k - для записи ключа к шифру* и *text - для записи сообщения которое нужно зашифровать*. Также для создания рандомного ключа ввожу функцию **Random** и использую randstring([‘ввожу переменные из которых делается выбор’], ‘задаю длину ключа’).



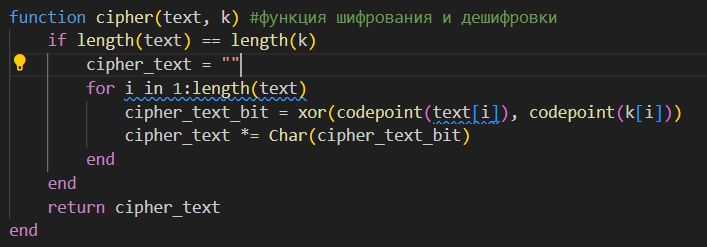
‘Введение функции и запись переменных’

Для шифрования нам нужно побитовое сложение произвожу его поэлементно при помощи цикла for. Само побитовое сложение выполняю при помощи функции xor. Также не забываем вернуть элементы в формат строки, использую для этого Char.



‘Создание цикла for с побитовым сложением’

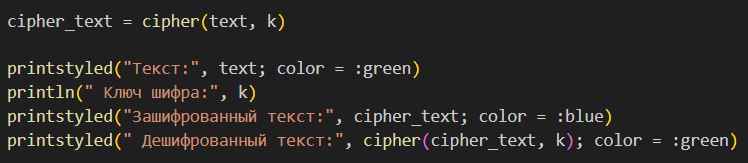
Не смотря на то, что создавая ключ через randstring, я задавала длину такую же как у введенного сообщения, для страховки создаю if, который проверяет условие равенста длин сообщения и ключа. Далее вписываю подготовленный цикл for, и добавляю return, чтобы функция возвращала зашифрованный текст.



‘Функция шифровки и дешифровки’

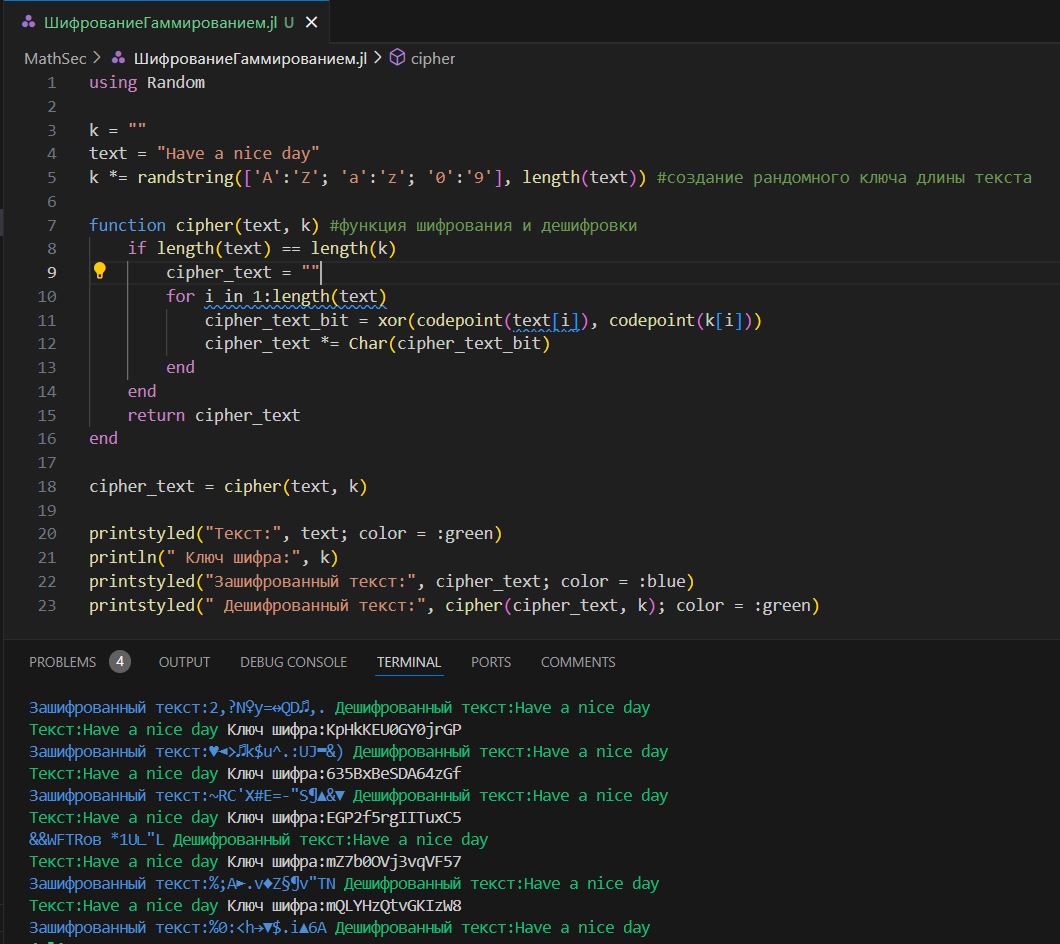
Для вывода переменных я использовала несколько разных способов: printstyled - позволяет меня цвет выводимой записи println - отвечает за перенос следующей записи на новую строку

В выводе дешифрованного текста я отправляю полученный зашифрованный текст обратно в созданную функцию с тем же ключом шифрования.



‘Вывод переменных’

Здесь представлен итоговый вид кода.



‘Итоговый код’

После запуска наблюдаю, что программа работает верно, цвета вводного текста и дешифрованного одинаковые и текст тоже одинаковый. Также выполняю несколько раз, чтобы убедиться, что рандомная генерация ключа работает и программа также работаем с любым из сгенерированных ключей.

‘Результат выполнения программы’

‘Результат выполнения программы’

# Выводы

В процессе выполнения работы я разобралась с принципом работы шифрования гаммированием. Изучила новые функции Julia, такие как randstring, xor и codepoint. Реализовала шифрование гаммированием на языке программирования Julia.

# Список литературы

::: Пособие по лабораторной работе 3